

# Computing Abelian regularities on RLE strings

Shiho Sugimoto<sup>1</sup>, Naoki Noda<sup>2</sup>, Shunsuke Inenaga<sup>1</sup>,  
Hideo Bannai<sup>1</sup>, and Masayuki Takeda<sup>1</sup>

<sup>1</sup> Department of Informatics, Kyushu University, Japan  
{shiho.sugimoto, inenaga, bannai, takeda}@inf.kyushu-u.ac.jp

<sup>2</sup> Department of Physics, Kyushu University, Japan

**Abstract.** Two strings  $x$  and  $y$  are said to be Abelian equivalent if  $x$  is a permutation of  $y$ , or vice versa. If a string  $z$  satisfies  $z = xy$  with  $x$  and  $y$  being Abelian equivalent, then  $z$  is said to be an *Abelian square*. If a string  $w$  can be factorized into a sequence  $v_1, \dots, v_s$  of strings such that  $v_1, \dots, v_{s-1}$  are all Abelian equivalent and  $v_s$  is a substring of a permutation of  $v_1$ , then  $w$  is said to have a *regular Abelian period*  $(p, t)$  where  $p = |v_1|$  and  $t = |v_s|$ . If a substring  $w_1[i..i+\ell-1]$  of a string  $w_1$  and a substring  $w_2[j..j+\ell-1]$  of another string  $w_2$  are Abelian equivalent, then the substrings are said to be a common Abelian factor of  $w_1$  and  $w_2$  and if the length  $\ell$  is the maximum of such then the substrings are said to be a *longest common Abelian factor* of  $w_1$  and  $w_2$ . We propose efficient algorithms which compute these Abelian regularities using the *run length encoding (RLE)* of strings. For a given string  $w$  of length  $n$  whose RLE is of size  $m$ , we propose algorithms which compute all Abelian squares occurring in  $w$  in  $O(mn)$  time, and all regular Abelian periods of  $w$  in  $O(mn)$  time. For two given strings  $w_1$  and  $w_2$  of total length  $n$  and of total RLE size  $m$ , we propose an algorithm which computes all longest common Abelian factors in  $O(m^2n)$  time.

## 1 Introduction

Two strings  $s_1$  and  $s_2$  are said to be *Abelian equivalent* if  $s_1$  is a permutation of  $s_2$ , or vice versa. For instance, strings *ababaac* and *caaabba* are Abelian equivalent. Since the seminal paper by Erdős [6] published in 1961, the study of Abelian equivalence on strings has attracted much attention, both in word combinatorics and string algorithmics.

### 1.1 Our problems and previous results

In this paper, we are interested in the following algorithmic problems related to Abelian regularities of strings.

1. Compute *Abelian squares* in a given string.
2. Compute *regular Abelian periods* of a given string.
3. Compute *longest common Abelian factors* of two given strings.

Cummings and Smyth [5] proposed an  $O(n^2)$ -time algorithm to solve Problem 1, where  $n$  is the length of the given string. Crochemore et al. [4] proposed an alternative  $O(n^2)$ -time solution to the same problem. Recently, Kociumaka et al. [12] showed how to compute all Abelian squares in  $O(s + \frac{n^2}{\log^2 n})$  time, where  $s$  is the number of outputs.

Related to Problem 2, various kinds of Abelian periods of strings have been considered: An integer  $p$  is said to be a *full Abelian period* of a string  $w$  iff there is a decomposition  $u_1, \dots, u_z$  of  $w$  such that  $|u_i| = p$  for all  $1 \leq i \leq z$  and  $u_1, \dots, u_z$  are all Abelian equivalent. A pair  $(p, t)$  of integers is said to be a *regular Abelian period* (or simply an *Abelian period*) of a string  $w$  iff there is a decomposition  $v_1, \dots, v_s$  of  $w$  such that  $p$  is a full Abelian period of  $v_1 \cdots v_{s-1}$ ,  $|v_i| = p$  for all  $1 \leq i \leq s-1$ , and  $v_s$  is a permutation of a substring of  $v_1$  (and hence  $t \leq p$ ). A triple  $(h, p, t)$  of integers is said to be a *weak Abelian period* of a string  $w$  iff there is a decomposition  $y_1, \dots, y_r$  of  $w$  such that  $(p, t)$  is an Abelian period of  $y_2 \cdots y_r$ ,  $|y_1| = h$ ,  $|y_i| = p$  for all  $2 \leq i \leq r-1$ ,  $|y_r| = t$ , and  $y_1$  is a permutation of a substring of  $y_2$  (and hence  $h \leq p$ ).

The study on Abelian periodicity of strings was initiated by Constantinescu and Ilie [3]. Fici et al. [8,9] gave an  $O(n \log \log n)$ -time algorithm to compute all full Abelian periods. Later, Kociumaka et al. [11] showed an optimal  $O(n)$ -time algorithm to compute all full Abelian periods. Fici et al. [8,9] also showed an  $O(n^2)$ -time algorithm to compute all regular Abelian periods for a given string of length  $n$ . Kociumaka et al. [11] also developed an algorithm which finds all regular Abelian periods in  $O(n(\log \log n + \log \sigma))$  time, where  $\sigma$  is the alphabet size. Fici et al. [7] proposed an algorithm which computes all weak Abelian periods in  $O(\sigma n^2)$  time, and later Crochemore et al. [4] proposed an improved  $O(n^2)$ -time algorithm to compute all weak Abelian periods. Kociumaka et al. [12] showed how to compute all *shortest* weak Abelian periods in  $O(n^2/\sqrt{\log n})$  time.

Consider two strings  $w_1$  and  $w_2$ . A pair  $(s_1, s_2)$  of a substring  $s_1$  of  $w_1$  and a substring  $s_2$  of  $w_2$  is said to be a *common Abelian factor* of  $w_1$  and  $w_2$ , iff  $s_1$  and  $s_2$  are Abelian equivalent. Alatabbi et al. [1] proposed an  $O(\sigma n^2)$ -time and  $O(\sigma n)$ -space algorithm to solve Problem 3 of computing all *longest common Abelian factors (LCAFs)* of two given strings of total length  $n$ . Later, Grabowski [10] showed an algorithm which finds all LCAFs in  $O(\sigma n^2)$  time with  $O(n)$  space. He also presented an  $O((\frac{\sigma}{k} + \log \sigma)n^2 \log n)$ -time  $O(kn)$ -space algorithm for a parameter  $k \leq \frac{\sigma}{\log \sigma}$ .

## 1.2 Our contribution

In this paper, we show that we can accelerate computation of Abelian regularities of strings via *run length encoding (RLE)* of strings. Namely, if  $m$  is the size of the RLE of a given string  $w$  of length  $n$ , we show that:

- (1) All Abelian squares in  $w$  can be computed in  $O(mn)$  time.
- (2) All regular Abelian periods of  $w$  can be computed in  $O(mn)$  time.

Since  $m \leq n$  always holds, our solution (1) is at least as efficient as the  $O(n^2)$ -time solutions by Cummings and Smyth [5] and by Crochemore et al. [4], and can

be much faster when the input string  $w$  is highly compressible by RLE. Amir et al. [2] proposed an  $O(\sigma(m^2 + n))$ -time algorithm to compute all Abelian squares using RLEs. Our  $O(mn)$ -time solution is faster than theirs when  $\frac{\sigma m^2}{m - \sigma} = o(n)$ . Our solution (2) is more efficient than the  $O(n(\log \log n + \log \sigma))$ -time solution by Kociumaka et al. [11] when  $\log \log n + \log \sigma = \omega(m)$ .

Also, if  $m$  is the total size of the RLEs of two given strings  $w_1$  and  $w_2$  of total length  $n$ , we show that:

- (3) All longest common Abelian factors of  $w_1$  and  $w_2$  can be computed in  $O(m^2 n)$  time.

Our solution (3) is more efficient than the  $O(\sigma n^2)$ -time solution by Grabowski [10] when  $\sigma n = \omega(m^2)$ .

All proofs omitted due to lack of space can be found in Appendix.

## 2 Preliminaries

Let  $\Sigma = \{c_1, \dots, c_\sigma\}$  be an ordered alphabet of size  $\sigma$ . An element of  $\Sigma^*$  is called a string. For any string  $w$ ,  $|w|$  denotes the length of  $w$ . The empty string is denoted by  $\varepsilon$ . Let  $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ . For any  $1 \leq i \leq |w|$ ,  $w[i]$  denotes the  $i$ -th symbol of  $w$ . For a string  $w = xyz$ , strings  $x$ ,  $y$ , and  $z$  are called a *prefix*, *substring*, and *suffix* of  $w$ , respectively. The substring of  $w$  that begins at position  $i$  and ends at position  $j$  is denoted by  $w[i..j]$  for  $1 \leq i \leq j \leq |w|$ . For convenience, let  $w[i..j] = \varepsilon$  for  $j > i$ .

For any string  $w \in \Sigma^*$ , its *Parikh vector*  $\mathcal{P}_w$  is an array of length  $\sigma$  such that for any  $1 \leq i \leq |\Sigma|$ ,  $\mathcal{P}_w[i]$  is the number of occurrences of each character  $c_i \in \Sigma$  in  $w$ . For example, for string  $w = \mathbf{abaab}$  over alphabet  $\Sigma = \{\mathbf{a}, \mathbf{b}\}$ ,  $\mathcal{P}_w = \langle 3, 2 \rangle$ . We say that strings  $x$  and  $y$  are *Abelian equivalent* if  $\mathcal{P}_x = \mathcal{P}_y$ . Note that  $\mathcal{P}_x = \mathcal{P}_y$  iff  $x$  and  $y$  are permutations of each other. When  $x$  is a substring of a permutation of  $y$ , then we write  $\mathcal{P}_x \subseteq \mathcal{P}_y$ . For any Parikh vectors  $P$  and  $Q$ , let  $\text{diff}(P, Q) = |\{i \mid P[i] \neq Q[i], 1 \leq i \leq \sigma\}|$ .

A non-empty string  $w$  of length  $2k$  is called an *Abelian square* if it is a concatenation of two Abelian equivalent strings of length  $k$  each, namely,  $\mathcal{P}_{w[1..k]} = \mathcal{P}_{w[k+1..2k]}$ . A string  $w$  is said to have a *regular Abelian period*  $(p, t)$  if  $w$  can be factorized into a sequence  $v_1, \dots, v_s$  of substrings such that  $p = |v_1| = \dots = |v_{s-1}|$ ,  $|v_s| = t$ ,  $\mathcal{P}_{v_i} = \mathcal{P}_{v_1}$  for all  $2 \leq i < s$ , and  $\mathcal{P}_{v_s} \subseteq \mathcal{P}_{v_1}$ . For any strings  $w_1, w_2 \in \Sigma^*$ , if a substring  $w_1[i..i + \ell - 1]$  of  $w_1$  and a substring  $w_2[j..j + \ell - 1]$  of  $w_2$  are Abelian equivalent, then the pair of substrings are said to be a *common Abelian factor* of  $w_1$  and  $w_2$ . When the length  $\ell$  is the maximum of such then the pair of substrings are said to be a *longest common Abelian factor* of  $w_1$  and  $w_2$ .

The *run length encoding* (RLE) of string  $w$  of length  $n$ , denoted  $RLE(w)$ , is a compact representation of  $w$  which encodes each maximal character run  $w[i..i + p - 1]$  by  $a^p$ , if  $w[j] = a$  for all  $i \leq j \leq i + p - 1$ , (2)  $w[i - 1] \neq w[i]$  or  $i = 1$ , and (3)  $w[i + p - 1] \neq w[i + p]$  or  $i + p - 1 = n$ . E.g.,  $RLE(\mathbf{aabb b b c c c c a a a \$}) = \mathbf{a^2 b^4 c^3 a^3 \$^1}$ . The *size* of  $RLE(T) = a_1^{p_1} \dots a_m^{p_m}$  is the number  $m$  of maximal

character runs in  $w$ , and each  $a_i^{p_i}$  is called an *RLE factor* of  $RLE(w)$ . Notice that  $m \leq n$  always holds. Also, since at most  $m$  distinct characters can appear in  $w$ , in what follows we will assume that  $\sigma \leq m$ . Even if the underlying alphabet is large, we can sort the characters appearing in  $w$  in  $O(m \log m)$  time and use this ordering in Parikh vectors. Since all of our algorithms will require at least  $O(mn)$  time, this  $O(m \log m)$ -time preprocessing is negligible.

For any  $1 \leq i \leq j \leq n$ , let  $RLE(w)[i..j] = a_i^{p_i} \cdots a_j^{p_j}$ . For convenience, let  $RLE(w)[i..j] = \varepsilon$  for  $i > j$ . For  $RLE(w) = a_1^{p_1} \cdots a_m^{p_m}$ , let  $RLE\_Bound(w) = \{1 + \sum_{i=1}^k p_i \mid 1 \leq k < m\} \cup \{1, n\}$ . For any  $1 \leq i \leq n$ , let  $succ(i) = \min\{j \in RLE\_Bound(w) \mid j > i\}$ . Namely,  $succ(i)$  is the smallest position in  $w$  that is greater than  $i$  and is either the beginning position of an RLE factor in  $w$  or the last position  $n$  in  $w$ .

### 3 Computing regular Abelian periods using RLEs

In this section, we propose an algorithm which computes all regular Abelian periods of a given string.

**Theorem 1.** *Given a string  $w$  of length  $n$  over an alphabet of size  $\sigma$ , we can compute all regular Abelian periods of  $w$  in  $O(mn)$  time and  $O(n)$  working space, where  $m$  is the size of  $RLE(w)$ .*

*Proof.* Our algorithm is very simple. Let  $n$  be the length of the input string  $w$  and  $m$  be the size of  $RLE(w)$ . We use a single window for each length  $d = 1, \dots, \lfloor \frac{n}{2} \rfloor$ .

For an arbitrarily fixed  $d$ , consider a decomposition  $v_1, \dots, v_s$  of  $w$  such that  $v_i = w[(i-1)d + 1..id]$  for  $1 \leq i \leq \lfloor \frac{n}{d} \rfloor$  and  $v_s = w[n - (n \bmod d) + 1..n]$ . Each  $v_i$  is called a block, and each block of length  $d$  is called a complete block.

There are two cases to consider.

- (a) If  $w$  is a unary string (i.e.,  $RLE(w) = a^n$  for some  $a \in \Sigma$ ). In this case,  $(d, (n \bmod d))$  is a regular Abelian period of  $w$  for any  $d$ . Also, note that this is the only case where  $(d, (n \bmod d))$  can be a regular Abelian period of any string of length  $n$  with  $RLE(v_i) = a^d$  for some complete block  $v_i$ . Clearly, it takes a total of  $O(n)$  time and  $O(1)$  space in this case.
- (b) If  $w$  contains at least two distinct characters, then observe that any complete block  $v_i$  is fully contained in a single RLE factor iff  $succ(1 + \sum_{k=1}^{i-1} |v_k|) = succ(\sum_{k=1}^i |v_k|)$ . Let  $S$  be an array of length  $n$  such that  $S[j] = succ(j)$  for each  $1 \leq j \leq n$ . We precompute this array  $S$  in  $O(n)$  time by a simple left-to-right scan over  $w$ . Using the precomputed array  $S$ , we can check in  $O(m)$  time if there exists a complete block  $v_i$  satisfying  $succ(1 + \sum_{k=1}^{i-1} |v_k|) = succ(\sum_{k=1}^i |v_k|)$ ; we process each complete block  $v_i$  in increasing order of  $i$  (from left to right), and stop as soon as we find the first complete block  $v_i$  with  $succ(1 + \sum_{k=1}^{i-1} |v_k|) = succ(\sum_{k=1}^i |v_k|)$ . If there exists such a complete block, then we can immediately determine that  $(d, (n \bmod d))$  is not a regular Abelian period (recall also Case (a) above.)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	a	b	b	a	a	a	b	a	b	a	a	a	a	b	b	a

**Fig. 1.**  $(3, 2)$  is a regular Abelian period of string  $w = aabbaaababaaaabbaa$  since  $\mathcal{P}_{w[1..3]} = \mathcal{P}_{w[4..6]} = \mathcal{P}_{w[7..9]} = \mathcal{P}_{w[10..12]} = \mathcal{P}_{w[13..15]} \supset \mathcal{P}_{w[16..17]}$ .

Now, assume that every complete block  $v_i$  spans at least two RLE factors. For each  $v_i$ , let  $m_i \geq 2$  be the number of RLE factors of  $RLE(w)$  that  $v_i$  spans (or alternatively,  $m_i$  is the size of  $RLE(v_i)$ ). We can compute  $\mathcal{P}_{v_i}$  in  $O(m_i)$  time from  $RLE(v_i)$ , by using the exponents of the elements of  $RLE(v_i)$ . Also, we can compare  $\mathcal{P}_{v_i}$  and  $\mathcal{P}_{v_{i-1}}$  in  $O(m_i)$  time, since there can be at most  $m_i$  distinct characters in  $v_i$  and hence it is enough to check the  $m_i$  entries of the Parikh vectors. Since there are  $\lfloor \frac{n}{d} \rfloor$  complete blocks and each complete block spans more than one RLE factor, we have  $\lfloor \frac{n}{d} \rfloor \leq \frac{1}{2} \sum_{i=1}^{s-1} m_i$ . Moreover, since each RLE factor is counted by a unique  $m_i$  or by a unique pair of  $m_{i-1}$  and  $m_i$  for some  $i$ , we have  $\sum_{i=1}^s m_i \leq 2m$ . Overall, it takes  $O(\sigma + \frac{n}{d} + \sum_{i=1}^s m_i) = O(m)$  time to determine whether or not  $(d, (n \bmod d))$  is a regular Abelian period of  $w$ . Consequently, it takes a total of  $O(mn)$  time to compute all regular Abelian periods of  $w$  for all  $d$ 's in this case. Since we use the array  $S$  of length  $n$  and we maintain two Parikh vectors of the two adjacent  $v_{i-1}$  and  $v_i$  for each  $i$ , the space requirement is  $O(\sigma + n) = O(n)$ .  $\square$

For example, let  $w = aabbaaababaaaabbaa$  and  $d = 3$ . See also Fig. 1 for illustration. We have  $RLE(w) = a^2b^2a^3b^1a^1b^1a^4b^2a^1$ . Then, we compute  $\mathcal{P}_{v_1} = \langle 2, 1 \rangle$  from  $RLE(v_1) = a^2b^1$ ,  $\mathcal{P}_{v_2} = \langle 2, 1 \rangle$  from  $RLE(v_2) = b^1a^2$ ,  $\mathcal{P}_{v_3} = \langle 2, 1 \rangle$  from  $RLE(v_3) = a^1b^1a^1$ ,  $\mathcal{P}_{v_4} = \langle 2, 1 \rangle$  from  $RLE(v_4) = b^1a^2$ ,  $\mathcal{P}_{v_5} = \langle 2, 1 \rangle$  from  $RLE(v_5) = a^2b^1$ , and  $\mathcal{P}_{v_6} = \langle 1, 1 \rangle$  from  $RLE(v_6) = b^1a^1$ . Since  $\mathcal{P}_{v_i} = \mathcal{P}_{v_1}$  for  $1 \leq i \leq 5$  and  $\mathcal{P}_{v_6} \subset \mathcal{P}_{v_1}$ ,  $(3, 2)$  is a regular Abelian period of the string  $w$ .

## 4 Computing Abelian squares using RLEs

In this section, we describe our algorithm to compute all Abelian squares occurring in a given string  $w$  of length  $n$ . Our algorithm is based on the algorithm of Cummings and Smyth [5] which compute all Abelian squares in  $w$  in  $O(n^2)$  time. We will improve the running time to  $O(mn)$ , where  $m$  is the size of  $RLE(w)$ .

### 4.1 Cummings and Smyth's $O(n^2)$ -time algorithm

Here we recall Cummings and Smyth's  $O(n^2)$ -time algorithm [5]. The algorithm is fairly straightforward: To compute Abelian squares in a given string  $w$ , their algorithm aligns two adjacent sliding windows of length  $d$  each, for every  $1 \leq d \leq \lfloor \frac{n}{2} \rfloor$ .

Consider an arbitrary fixed  $d$ . For each position  $1 \leq i \leq n - 2d + 1$  in  $w$ , let  $L_i$  and  $R_i$  denote the left and right windows aligned at position  $i$ . Namely,  $L_i = w[i..i + d - 1]$  and  $R_i = w[i + d..i + 2d - 1]$ . At the beginning of, the algorithm computes  $\mathcal{P}_{L_1}$  and  $\mathcal{P}_{R_1}$  for position 1 in  $w$ . It takes  $O(d)$  time to compute these Parikh vectors and  $O(\sigma)$  time compute  $\text{diff}(\mathcal{P}_{L_1}, \mathcal{P}_{R_1})$ . Assume  $\mathcal{P}_{L_i}$ ,  $\mathcal{P}_{R_i}$ , and  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i})$  have been computed for position  $i \geq 1$ , and  $\mathcal{P}_{L_{i+1}}$ ,  $\mathcal{P}_{R_{i+1}}$ , and  $\text{diff}(\mathcal{P}_{L_{i+1}}, \mathcal{P}_{R_{i+1}})$  is to be computed for the next position  $i + 1$ . A key observation is that given  $\mathcal{P}_{L_i}$ , then  $\mathcal{P}_{L_{i+1}}$  for the left window  $L_{i+1}$  for the next position  $i + 1$  can be easily computed in  $O(1)$  time, since at most two entries of the Parikh vector can change. The same applies to  $\mathcal{P}_{R_i}$  and  $\mathcal{P}_{R_{i+1}}$ . Also, given  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i})$  for the two adjacent windows  $L_i$  and  $R_i$  for position  $i$ , then it takes  $O(1)$  time to determine whether or not  $\text{diff}(\mathcal{P}_{L_{i+1}}, \mathcal{P}_{R_{i+1}}) = 0$  for the two adjacent windows  $L_{i+1}$  and  $R_{i+1}$  for the next position  $i + 1$ . Hence, for each  $d$ , it takes  $O(n)$  time to find all Abelian squares of length  $2d$ , and thus it takes a total of  $O(n^2)$  time for all  $1 \leq d \leq \lfloor \frac{n}{2} \rfloor$ .

## 4.2 Our $O(mn)$ -time algorithm

We propose an algorithm which computes all Abelian squares in a given string  $w$  of length  $n$  in  $O(mn)$  time, where  $m$  is the size of  $RLE(w)$ .

Our algorithm will output consecutive Abelian squares  $w[i..i + 2d - 1]$ ,  $w[i + 1..i + 2d]$ ,  $\dots$ ,  $w[j..j + 2d - 1]$  of length  $2d$  each as a triple  $\langle i, j, d \rangle$ . A single Abelian square  $w[i..i + 2d - 1]$  of length  $2d$  will be represented by  $\langle i, i, d \rangle$ .

For any position  $i$  in  $w$ , let  $\text{beg}(L_i)$  and  $\text{end}(L_i)$  respectively denote the beginning and ending positions of the left window  $L_i$ , and let  $\text{beg}(R_i)$  and  $\text{end}(R_i)$  respectively denote the beginning and ending positions of the right window  $R_i$ . Namely,  $\text{beg}(L_i) = i$ ,  $\text{end}(L_i) = i + d - 1$ ,  $\text{beg}(R_i) = i + d$ , and  $\text{end}(R_i) = i + 2d - 1$ . Cummings and Smyth's algorithm described above increases each of  $\text{beg}(L_i)$ ,  $\text{end}(L_i)$ ,  $\text{beg}(R_i)$ , and  $\text{end}(R_i)$  one by one, and tests all positions  $i = 1, \dots, n - 2d + 1$  in  $w$ . Hence their algorithm takes  $O(n)$  time for each window size  $d$ .

In what follows, we show that it is indeed enough to check only  $O(m)$  positions in  $w$  for each window size  $d$ . The outline of our algorithm is as follows. As Cummings and Smyth's algorithm, we use two adjacent windows of size  $d$ , and slide the windows. However, unlike Cummings and Smyth's algorithm where the windows are shifted by one position, in our algorithm the windows can be shifted by more than one position. The positions that are not skipped and are explicitly examined will be characterized by the RLE of  $w$ , and the equivalence of the Parikh vectors of the two adjacent windows for the skipped positions can easily be checked by simple arithmetics.

Now we describe our algorithm in detail. First, we compute  $RLE(w)$  and let  $m$  be its size. Consider an arbitrarily fixed window length  $d \geq 1$ .

**Initial step for position 1.** Initially, we compute  $\mathcal{P}_{L_1}$  and  $\mathcal{P}_{R_1}$  for position 1. We can compute these Parikh vectors in  $O(m)$  time and  $O(\sigma)$  space using the same method as in the algorithm of Theorem 1 in Section 3.

**Steps for positions larger than 1.** For each position  $i \geq 1$  in a given string  $w$ , let  $D_1^i = \text{succ}(\text{beg}(L_i)) - i$ ,  $D_2^i = \text{succ}(\text{end}(L_i)) - i$ , and  $D_3^i = \text{succ}(\text{end}(R_i)) - i$ . The *break point* for each position  $i$ , denoted  $\text{bp}(i)$ , is defined by  $i + \min\{D_1^i, D_2^i, D_3^i\}$ . Assume the left window is aligned at position  $i$  in  $w$ . Then, we leap to the break point  $\text{bp}(i)$  directly from  $i$ . In other words, the two windows  $L_i$  and  $R_i$  are directly shifted to  $L_{\text{bp}(i)}$  and  $R_{\text{bp}(i)}$ , respectively.

It depends on the value of  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i})$  whether there can be an Abelian square between positions  $i$  and  $\text{bp}(i)$ . Note that  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) \neq 1$ . Below, we characterize the other cases in detail.

**Lemma 1.** Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 0$ . Then, for any  $i < j \leq \text{bp}(i)$ ,  $j$  is the beginning position of an Abelian square of length  $2d$  iff  $w[\text{beg}(L_i)] = w[\text{beg}(R_i)] = w[\text{end}(R_i) + 1]$ .

**Lemma 2.** Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 2$ . Let  $c_p$  be the unique character which occurs more on the left window  $L_i$  than on the right window  $R_i$ , and  $c_q$  be the unique character which occurs more on the right window  $R_i$  than on the left window  $L_i$ . Let  $x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] > 0$ , and assume  $x \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Then,  $i + x$  is the beginning position of an Abelian square of length  $2d$  iff  $w[\text{beg}(L_i)] = c_p$ ,  $w[\text{beg}(R_i)] = c_q = w[\text{end}(R_i) + 1]$ . Also, this is the only Abelian square of length  $2d$  beginning at positions between  $i$  and  $\text{bp}(i)$ .

**Lemma 3.** Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 2$ . Let  $c_p$  be the unique character which occurs more on the left window  $L_i$  than on the right window  $R_i$ , and  $c_q$  be the unique character which occurs more on the right window  $R_i$  than on the left window  $L_i$ . Let  $x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] > 0$ , and assume  $\frac{x}{2} \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Then,  $i + \frac{x}{2}$  is the beginning position of an Abelian square of length  $2d$  iff  $w[\text{beg}(L_i)] = c_p = w[\text{end}(R_i) + 1]$ ,  $w[\text{beg}(R_i)] = c_q$ . Also, this is the only Abelian square of length  $2d$  beginning at positions between  $i$  and  $\text{bp}(i)$ .

**Lemma 4.** Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 3$ . Let  $c_p = w[\text{beg}(L_i)]$ ,  $c_{p'} = w[\text{end}(R_i) + 1]$ , and  $c_q = w[\text{beg}(R_i)]$ . Then,  $i + x$  with  $i < i + x \leq \text{bp}(i)$  is the beginning position of an Abelian square of length  $2d$  iff  $0 < x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{L_i}[p'] - \mathcal{P}_{R_i}[p'] = \frac{\mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q]}{2} \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Also, this is the only Abelian square of length  $2d$  beginning at positions between  $i$  and  $\text{bp}(i)$ .

**Lemma 5.** Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) \geq 4$ . Then, there exists no Abelian square of length  $2d$  beginning at any position  $j$  with  $i < j \leq \text{bp}(i)$ .

**Main result.** We are ready to show the main result of this section.

**Theorem 2.** Given a string  $w$  of the length  $n$  over an alphabet of size  $\sigma$ , we can compute all Abelian squares in  $w$  in  $O(mn)$  time and  $O(n)$  working space, where  $m$  is the size of  $\text{RLE}(w)$ .

*Proof.* Consider an arbitrarily fixed window length  $d$ . As was explained, it takes  $O(m)$  time to compute  $\mathcal{P}_{L_1}$ ,  $\mathcal{P}_{R_1}$ , and  $\text{diff}(\mathcal{P}_{L_1}, \mathcal{P}_{R_1})$  for the initial position



1. Suppose that the two windows are aligned at some position  $i \geq 1$ . Then, our algorithm computes Abelian squares starting at positions between  $i$  and  $\text{bp}(i)$  using one of Lemma 1, Lemma 2, Lemma 3, Lemma 4, and Lemma 5, depending on the value of  $\text{diff}(\mathcal{P}_{L_1}, \mathcal{P}_{R_i})$ . In each case, all Abelian squares of length  $2d$  starting at positions between  $i$  and  $\text{bp}(i)$  can be computed in  $O(1)$  time by simple arithmetics. Then, the left and right windows  $L_i$  and  $R_i$  are shifted to  $L_{\text{bp}(i)}$  and  $R_{\text{bp}(i)}$ , respectively. Using the array  $S$  as in Theorem 1, we can compute  $\text{bp}(i)$  in  $O(1)$  time for a given position  $i$  in  $w$ .

Let us analyze the number of times the windows are shifted for each  $d$ . Since  $\text{bp}(i) = i + \min\{D_1^i, D_2^i, D_3^i\}$ , for each position  $p$  there can be at most three distinct positions  $i, j, k$  such that  $p = \text{bp}(i) = \text{bp}(j) = \text{bp}(k)$ . Thus, for each  $d$  we shift the two adjacent windows at most  $3m$  times.

Overall, our algorithm runs in  $O(mn)$  time for all window lengths  $d = 1, \dots, \lfloor n/2 \rfloor$ . The space requirement is  $O(n)$  since we need to maintain the Parikh vectors of the two sliding windows and the array  $S$ .  $\square$

Example on how our algorithm computes all Abelian squares using RLEs can be found in Appendix B.1.

## 5 Computing longest common Abelian factors using RLEs

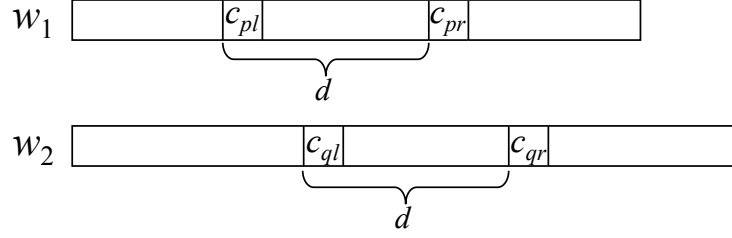
In this section, we introduce our RLE-based algorithm which computes longest common Abelian factors of two given strings  $w_1$  and  $w_2$ . Formally, we solve the following problem. Let  $n = \min\{|w_1|, |w_2|\}$ . Given two strings  $w_1$  and  $w_2$ , compute the length  $l = \max\{d \mid \mathcal{P}_{w_1[i..i+d-1]} = \mathcal{P}_{w_2[k..k+d-1]}, 1 \leq d \leq n\}$  of the longest common Abelian factor(s) of  $w_1$  and  $w_2$ , together with a pair  $(i, j)$  of positions on  $w_1$  and  $w_2$  such that  $\mathcal{P}_{w_1[i..i+l-1]} = \mathcal{P}_{w_2[k..k+l-1]}$ .

Our algorithm uses an idea from Alattabi et al.'s algorithm [1]. For each window size  $d$ , their algorithm computes the Parikh vectors of all substrings of  $w_1$  and  $w_2$  of length  $d$  in  $O(\sigma n)$  time, using two windows of length  $d$  each. Then they sort the Parikh vectors in  $O(\sigma n)$  time, and output the largest  $d$  for which common Parikh vectors exist for  $w_1$  and  $w_2$ , together with the lists of respective occurrences of longest common Abelian factors. The total time requirement is clearly  $O(\sigma n^2)$ .

Our algorithm is different from Alattabi et al.'s algorithm in that (1) we use RLEs of strings  $w_1$  and  $w_2$  and (2) we avoid to sort the Parikh vectors. As in the previous sections, for a given window length  $d$  ( $1 \leq n$ ), we shift two windows of length  $d$  over both of  $RLE(w_1)$  and  $RLE(w_2)$ , and stops when we reach a break point of  $RLE(w_1)$  or  $RLE(w_2)$ . We then check if there is a common Abelian factors in the ranges of  $w_1$  and  $w_2$  we are looking at.

Since we use a single window for each of the input strings  $w_1$  and  $w_2$ , we need to modify the definition of the break points. Let  $U_i$  and  $V_k$  be the sliding windows for  $w_1$  and  $w_2$  that are aligned at position  $i$  of  $w_1$  and at position  $k$  of  $w_2$ , respectively. For each position  $i \geq 1$  in  $w_1$ , let  $\text{bp}_1(i) = i + \min\{D_1^i, D_2^i\}$ ,





**Fig. 2.** Conceptual drawing of  $c_{pl}$ ,  $c_{pr}$ ,  $c_{qr}$ , and  $c_{ql}$ .

where  $D_1^i = \text{succ}(\text{beg}(U_i)) - i$  and  $D_2^i = \text{succ}(\text{end}(U_i)) - i$ . For each position  $k \geq 1$  in  $w_2$ ,  $\text{bp}_2(k)$  is defined analogously.

Consider an arbitrarily fixed window length  $d$ . Assume that we have just shifted the window on  $w_1$  from position  $i$  (i.e.,  $V_i = w_1[i..i + d - 1]$ ) to the break point  $\text{bp}_1(i)$  (i.e.,  $V_{\text{bp}_1(i)} = w_1[\text{bp}_1(i)..\text{bp}_1(i) + d - 1]$ ). Let  $c_{pl} = w_1[i]$  and  $c_{pr} = w_1[i + d]$  (see also Fig. 2).

For characters  $c_{pl}$  and  $c_{pr}$ , we consider the minimum and maximum numbers of occurrences of these characters during the slide from position  $i$  to  $\text{bp}_1(i)$ . Let  $\min(p_l) = \mathcal{P}_{w_1[\text{bp}_1(i)..\text{bp}_1(i) + d - 1]}[p_l]$ ,  $\max(p_l) = \mathcal{P}_{w_1[i..i + d - 1]}[p_l]$ ,  $\min(p_r) = \mathcal{P}_{w_1[i..i + d - 1]}[p_r]$  and  $\max(p_r) = \mathcal{P}_{w_1[\text{bp}_1(i)..\text{bp}_1(i) + d - 1]}[p_r]$ . We will use these values to determine if there is a common Abelian factor of length  $d$  for  $w_1$  and  $w_2$ .

Also, assume that we have just shifted the window on  $w_2$  from position  $k$  (i.e.,  $U_k = w_2[k..k + d - 1]$ ) to the break point  $\text{bp}_2(k)$  (i.e.,  $U_{\text{bp}_2(k)} = w_2[\text{bp}_2(k)..\text{bp}_2(k) + d - 1]$ ). Let  $c_{ql} = w_2[k]$  and  $c_{qr} = w_2[k + d]$  (see also Fig. 2). For characters  $c_{ql}$  and  $c_{qr}$ , we also consider the minimum and maximum numbers of occurrences of these characters during the slide from position  $k$  to  $\text{bp}_2(k)$ . Let  $\min(q_l) = \mathcal{P}_{w_2[\text{bp}_2(k)..\text{bp}_2(k) + d - 1]}[q_l]$ ,  $\max(q_l) = \mathcal{P}_{w_2[k..k + d - 1]}[q_l]$ ,  $\min(q_r) = \mathcal{P}_{w_2[k..k + d - 1]}[q_r]$  and  $\max(q_r) = \mathcal{P}_{w_2[\text{bp}_2(k)..\text{bp}_2(k) + d - 1]}[q_r]$ .

Let  $m$  be the total size of  $RLE(w_1)$  and  $RLE(w_2)$ , and  $l$  be the length of longest common Abelian factors of  $w_1$  and  $w_2$ . Our algorithm computes an  $O(m^2)$ -size representation of every pair  $(i, k)$  of positions for which  $(w_1[i..i + l - 1], w_2[k..k + l - 1])$  is a longest common Abelian factor of  $w_1$  and  $w_2$ .

In the lemmas which follow, we assume that  $\mathcal{P}_{w_1[i..i + d - 1]}[v] = \mathcal{P}_{w_2[k..k + d - 1]}[v]$  for any  $v \in \{1, \dots, \sigma\} \setminus \{p_l, p_r, q_l, q_r\}$ . This is because, if this condition is not satisfied, then there cannot be an Abelian common factor of length  $d$  for positions between  $i$  to  $\text{bp}_1(i)$  in  $w_1$  and position between  $k$  to  $\text{bp}_2(k)$  in  $w_2$ .

**Lemma 6.** *Assume  $c_{pl} = c_{pr}$  and  $c_{ql} = c_{qr}$ . Then, for any pair of positions  $i \leq i' \leq \text{bp}_1(i)$  and  $k \leq k' \leq \text{bp}_2(k)$ ,  $(w_1[i'..i' + d - 1], w_2[k'..k' + d - 1])$  is an Abelian common factor of length  $d$  iff  $\mathcal{P}_{w_1[i..i + d - 1]} = \mathcal{P}_{w_2[k..k + d - 1]}$ .*

*Proof.* Since  $c_{p_l} = c_{p_r}$  and  $c_{q_l} = c_{q_r}$ , the Parikh vectors of the sliding windows do not change during the slides from  $i$  to  $\text{bp}_1(i)$  and from  $k$  to  $\text{bp}_2(k)$ . Thus the lemme holds.  $\square$

**Lemma 7.** Assume  $c_{p_l} = c_{q_l} \neq c_{p_r} = c_{q_r}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $0 \leq x \leq \text{bp}_1(i) - i$ ,  $0 \leq y \leq \text{bp}_2(k) - k$  and  $x - y = \max(p_l) - \max(q_l) = \min(q_r) - \min(p_r)$ .

**Lemma 8.** Assume  $c_{p_r} \neq c_{p_l} = c_{q_l} \neq c_{q_r}$  and  $c_{p_r} \neq c_{q_r}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \geq 0$ ,  $y = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \geq 0$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y$ .

**Lemma 9.** Assume  $c_{p_l} \neq c_{p_r} = c_{q_r} \neq c_{q_l}$  and  $c_{p_l} \neq c_{q_l}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] \geq 0$ ,  $y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] \geq 0$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y$ .

**Lemma 10.** Assume  $c_{p_l} = c_{q_r} \neq c_{p_r} = c_{q_l}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $x + y = \min(p_r) - \max(q_l) = \max(q_l) - \min(p_r)$ ,  $0 \leq x \leq \text{bp}_1(i) - i$  and  $0 \leq y \leq \text{bp}_2(k) - k$ .

**Lemma 11.** Assume  $c_{p_l}, c_{p_r}, c_{q_l}$  and  $c_{q_r}$  are mutually distinct. There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $0 \leq x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \leq \text{bp}_1(i) - i$  and  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$ .

**Lemma 12.** Assume  $c_{q_l} \neq c_{p_l} = c_{p_r} \neq c_{q_r}$  and  $c_{q_l} \neq c_{q_r}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $0 \leq x \leq \text{bp}_1(i) - i$ ,  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_l]$ .

**Lemma 13.** Assume  $c_{p_l} \neq c_{q_l} = c_{q_r} \neq c_{p_r}$  and  $c_{p_l} \neq c_{p_r}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $0 \leq y \leq \text{bp}_2(k) - k$  and  $x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \geq 0$ .

**Lemma 14.** Assume  $c_{p_r} \neq c_{p_l} = c_{q_r} \neq c_{q_l}$  and  $c_{p_r} \neq c_{q_l}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $0 \leq x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \leq \text{bp}_1(i) - i$ ,  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] \leq \text{bp}_2(k) - k$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y$ .

**Lemma 15.** Assume  $c_{p_l} \neq c_{q_l} = c_{p_r} \neq c_{q_r}$  and  $c_{p_l} \neq c_{q_r}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$  iff  $0 \leq x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] \leq \text{bp}_1(i) - i$ ,  $0 \leq y = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y$ .

**Theorem 3.** Given two strings  $w_1$  and  $w_2$ , we can compute an  $O(m^2)$ -size representation of all longest common Abelian factors of  $w_1$  and  $w_2$  in  $O(m^2n)$  time with  $O(\sigma)$  working space, where  $m$  and  $n$  are the total size of the RLEs and the total length of  $w_1$  and  $w_2$ , respectively.

*Proof.* The correctness should be clear from Lemmas 6–15.

Let  $m_1, m_2$  be the sizes of  $RLE(w_1)$  and  $RLE(w_2)$ , respectively. Let  $n_{\min} = \min\{|w_1|, |w_2|\}$ . For each fixed window size  $d$ , the window for  $w_1$  shifts over  $w_1$   $O(m_1)$  times. For each shift of the window for  $w_1$ , the window for  $w_2$  shifts over  $w_2$   $O(m_2)$  times. Thus, we have  $O(m_1 \cdot m_2 \cdot n_{\min})$  total shifts. Since all the conditions in Lemmas 6–15 can be tested in  $O(1)$  time each by simple arithmetic, the total time complexity is  $O(m_1 m_2 n_{\min} + n)$ , where the  $n$  term denotes the cost to compute  $RLE(w_1)$  and  $RLE(w_2)$ . Thus, it is clearly bounded by  $O(m^2 n)$ . Next, we focus on the output size. Let  $l$  be the length of the longest common Abelian factors of  $w_1$  and  $w_2$ . Using Lemmas 7–15, for each pair of the shifts of the two windows we can compute an  $O(1)$ -size representation of the longest common Abelian factors found. Since there are  $O(m_1 \cdot m_2)$  shifts for window length  $l$ , the output size is bounded by  $O(m^2)$ . The working space is  $O(\sigma)$ , since we only need to maintain two Parikh vectors for the two sliding windows.  $\square$

**Examples.** We show an example of how our algorithm compute a common Abelian factor of length 4 for two input strings  $w_1 = aaaaacbbbcc$  and  $w_2 = cccaaccbbb$ .

$$\begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ w_1 : & a & a & \boxed{a} & \boxed{a} & \boxed{a} & \boxed{c} & b & b & b & c & c \end{array}$$

$$w_2 : \boxed{c} \boxed{c} \boxed{c} \boxed{a} \boxed{a} c c b b b b$$

**Fig. 3.** Showing two sliding window of length  $d = 4$ , where  $i = 3$ ,  $bp_1(i) = 6$ ,  $k = 1$ ,  $bp_2(k) = 2$ ,  $c_{p_l} = a$ ,  $c_{p_r} = b$ ,  $c_{q_l} = c$ ,  $c_{q_r} = a$ .

$$\begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ w_1 : & a & a & \boxed{a} & \boxed{a} & \boxed{a} & \boxed{c} & b & b & b & c & c \end{array}$$

$$w_2 : c \boxed{c} \boxed{c} \boxed{a} \boxed{a} \boxed{c} \boxed{c} b b b b$$

**Fig. 4.** Showing two sliding windows of length  $d = 4$ , where  $i = 3$ ,  $bp_1(i) = 6$ ,  $k = 2$ ,  $bp_2(k) = 4$ ,  $c_{p_l} = a$ ,  $c_{p_r} = b$ ,  $c_{q_l} = c$ ,  $c_{q_r} = c$ .

Suppose that the window for  $w_1$  is now aligned at position 3 of  $w_1$  (namely  $U_3 = w_1[3..6] = aaac$ ). We then shift it to position  $bp_1(3) = 6$  (namely  $U_6 = w_1[6..9] = cbbb$ ). For this shift of the window on  $w_1$ , we test  $O(m_2)$  shifts of the window over the second string  $w_2$ , as follows.

We begin with position 1 of the other string  $w_2$  (namely  $V_1 = w_2[1..4] = ccca$ ), and shift the window to position  $bp_2(1) = 2$ . See also Fig. 3. It follows from Lemma 14 that there is no common Abelian factor during these slides. We move on to the next step.

Next, the window for  $w_2$  is shifted from position 2 to position  $bp_2(2) = 4$  (namely,  $V_4 = w_2[4..7] = aacc$ ). See also Fig. 4. It follows from Lemma 13 that there is no common Abelian factor during the slides. We move on to the next step.

Next, the window for  $w_2$  is shifted from position 4 to position  $bp_2(4) = 6$  (namely,  $V_6 = w_2[6..9] = cccb$ ). See also Fig. 5. Since the numbers of occurrences

$$w_1 : a a \overline{a a a c} \overline{b b b} c c$$

$$w_2 : c c c \overline{a a c c} \overline{b b} b b$$

**Fig. 5.** Showing two sliding windows of length  $d = 4$ , where  $i = 3$ ,  $\text{bp}_1(i) = 3$ ,  $k = 4$ ,  $\text{bp}_2(k) = 6$ ,  $c_{p_l} = a$ ,  $c_{p_r} = b$ ,  $c_{q_l} = a$ ,  $c_{q_r} = b$ .

$$w_1 : a a \overline{a a a c} \overline{b b b} c c$$

$$w_2 : c c c a \overline{c c} \overline{b b} b b$$

**Fig. 6.** Showing two sliding windows of length  $d = 4$ , where  $i = 3$ ,  $\text{bp}_1(i) = 3$ ,  $k = 6$ ,  $\text{bp}_2(k) = 3$ ,  $c_{p_l} = a$ ,  $c_{p_r} = b$ ,  $c_{q_l} = c$ ,  $c_{q_r} = b$ .

of  $c$  on  $w_1$  and  $w_2$  are different and  $c$  is not equal to  $a$  or  $b$ , there is no common Abelian factor during the slides. We move on to the next step.

Next, the window for  $w_2$  is shifted from position 6 to position  $\text{bp}_2(6) = 8$ . See Fig. 5. It follows from Lemma 9 that there is a common Abelian factor  $(w_1[6..9], w_2[7..10])$  of length  $d = 4$ .

## References

1. A. Alatabbi, C. S. Iliopoulos, A. Langiu, and M. S. Rahman. Algorithms for longest common Abelian factors. <http://arxiv.org/abs/1503.00049>.
2. A. Amir, A. Apostolico, T. Hirst, G. M. Landau, N. Lewenstein, and L. Rozenberg. Algorithms for jumbled indexing, jumbled border and jumbled square on run-length encoded strings. In *SPIRE 2014*, pages 45–51, 2014.
3. S. Constantinescu and L. Ilie. Fine and Wilf’s theorem for Abelian periods. *Bulletin of the EATCS*, 89:167–170, 2006.
4. M. Crochemore, C. S. Iliopoulos, T. Kociumaka, M. Kubica, J. Pachocki, J. Radoszewski, W. Rytter, W. Tyczyński, and T. Waleń. A note on efficient computation of all Abelian periods in a string. *Inf. Process. Lett.*, 113(3):74–77, 2013.
5. L. J. Cummings and W. F. Smyth. Weak repetitions in strings. *J. Combinatorial Mathematics and Combinatorial Computing*, 24:33–48, 1997.
6. P. Erdős. Some unsolved problems. *Hungarian Academy of Sciences Mat. Kutató Intézet Közl.*, 6:221–254, 1961.
7. G. Fici, T. Lecroq, A. Lefebvre, and É. Prieur-Gaston. Online computation of Abelian runs. *CoRR*, abs/1501.01429, 2015.
8. G. Fici, T. Lecroq, A. Lefebvre, É. Prieur-Gaston, and W. F. Smyth. Quasi-linear time computation of the Abelian periods of a word. In *PSC 2012*, pages 103–110, 2012.
9. G. Fici, T. Lecroq, A. Lefebvre, É. Prieur-Gaston, and W. F. Smyth. A note on easy and efficient computation of full Abelian periods of a word. *CoRR*, abs/1510.00634, 2015.
10. S. Grabowski. A note on the longest common Abelian factor problem. *CoRR*, abs/1503.01093, 2015.
11. T. Kociumaka, J. Radoszewski, and W. Rytter. Fast algorithms for Abelian periods in words and greatest common divisor queries. In *STACS 2013*, pages 245–256, 2013.
12. T. Kociumaka, J. Radoszewski, and B. Wisniewski. Subquadratic-time algorithms for Abelian stringology problems. In *MACIS 2015*, pages 320–334, 2015.

## A Appendix: Proofs

In this appendix, we provide proofs which are omitted due to lack of space.

### A.1 Proofs for lemmas in Section 3

**Lemma 1.** *Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 0$ . Then, for any  $i < j \leq \text{bp}(i)$ ,  $j$  is the beginning position of an Abelian square of length  $2d$  iff  $w[\text{beg}(L_i)] = w[\text{beg}(R_i)] = w[\text{end}(R_i) + 1]$ .*

*Proof.* ( $\Leftarrow$ ) By the definition of  $\text{bp}(i)$ ,  $w[\text{beg}(L_i)] = w[\text{beg}(L_j)]$ ,  $w[\text{beg}(R_i)] = w[\text{beg}(R_j)]$ , and  $w[\text{end}(R_i) + 1] = w[\text{end}(R_j) + 1]$  for all  $i < j \leq \text{bp}(i)$ . Let  $c = w[\text{beg}(L_i)] = w[\text{beg}(R_i)] = w[\text{end}(R_i) + 1]$ . Then we have  $w[\text{beg}(L_j)] = w[\text{beg}(R_j)] = w[\text{end}(R_j) + 1] = c$ . Thus the Parikh vectors of the sliding windows do not change at any position between  $i$  and  $\text{bp}(i)$ . Since we have assumed  $\mathcal{P}_{L_i} = \mathcal{P}_{R_i}$ ,  $\mathcal{P}_{L_j} = \mathcal{P}_{R_j}$  for any  $i < j \leq \text{bp}(i)$ . Thus  $w[j..j + 2d - 1] = L_j R_j$  is an Abelian square of length  $2d$  for any  $i < j \leq \text{bp}(i)$ .

( $\Rightarrow$ ) Since  $j$  is the beginning position of an Abelian square of length  $2d$ ,  $\mathcal{P}_{L_j} = \mathcal{P}_{R_j}$ . Let  $c_p = w[\text{beg}(L_i)]$ ,  $c_q = w[\text{beg}(R_i)]$ , and  $c_t = w[\text{end}(R_i) + 1]$ . By the definition of  $\text{bp}(i)$ ,  $w[\text{beg}(L_j)] = c_p$ ,  $w[\text{beg}(R_j)] = c_q$ , and  $w[\text{end}(R_j) + 1] = c_t$  for any  $i < j \leq \text{bp}(i)$ . Also, for any  $i < j \leq \text{bp}(i)$ ,  $\mathcal{P}_{L_j}[x] = \mathcal{P}_{L_i}[x] - j + i$ ,  $\mathcal{P}_{L_j}[y] = \mathcal{P}_{L_i}[y] + j - i$ ,  $\mathcal{P}_{R_j}[y] = \mathcal{P}_{R_i}[y] - j + i$ , and  $\mathcal{P}_{R_j}[z] = \mathcal{P}_{R_i}[z] + j - i$ . Recall we have assumed that  $\mathcal{P}_{L_i} = \mathcal{P}_{R_i}$  and  $\mathcal{P}_{L_j} = \mathcal{P}_{R_j}$  for any  $i < j \leq \text{bp}(i)$ . This is possible only if  $c_p = c_q = c_t$ , namely,  $w[\text{beg}(L_j)] = w[\text{beg}(R_j)] = w[\text{end}(R_j) + 1]$ .

**Lemma 2.** *Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 2$ . Let  $c_p$  be the unique character which occurs more on the left window  $L_i$  than on the right window  $R_i$ , and  $c_q$  be the unique character which occurs more on the right window  $R_i$  than on the left window  $L_i$ . Let  $x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] > 0$ , and assume  $x \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Then,  $i + x$  is the beginning position of an Abelian square of length  $2d$  iff  $w[\text{beg}(L_i)] = c_p$ ,  $w[\text{beg}(R_i)] = c_q = w[\text{end}(R_i) + 1]$ . Also, this is the only Abelian square of length  $2d$  beginning at positions between  $i$  and  $\text{bp}(i)$ .*

*Proof.* ( $\Leftarrow$ ) Since  $w[\text{beg}(L_i)] = c_p$  and  $w[\text{beg}(R_i)] = w[\text{end}(R_i) + 1] = c_q$ , we have that  $\mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] - z = \mathcal{P}_{L_{i+z}}[p] - \mathcal{P}_{R_{i+z}}[p]$  and  $\mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] + z = \mathcal{P}_{R_{i+z}}[q] - \mathcal{P}_{L_{i+z}}[q]$  for any  $1 \leq z \leq \min\{D_1^i, D_2^i, D_3^i\}$ . By the definition of  $x$ , the Parikh vectors of the sliding windows become equal at position  $i + x$ .

( $\Rightarrow$ ) Since  $x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] > 0$ ,  $\mathcal{P}_{L_{i+x}}[p] = \mathcal{P}_{L_{i+x}}[p]$ , and  $\mathcal{P}_{L_{i+x}}[q] = \mathcal{P}_{L_{i+x}}[q]$ , we have  $w[\text{beg}(L_i)] = c_p$  and  $w[\text{beg}(R_i)] = w[\text{end}(R_i) + 1] = c_q$ .

From the above arguments, it is clear that  $i + x$  is the only position between  $i$  and  $\text{bp}(i)$  where an Abelian square of length  $2d$  can start.  $\square$

**Lemma 3.** *Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 2$ . Let  $c_p$  be the unique character which occurs more on the left window  $L_i$  than on the right window  $R_i$ , and  $c_q$  be the unique character which occurs more on the right window  $R_i$  than on the left window  $L_i$ . Let  $x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] > 0$ , and assume*

$\frac{x}{2} \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Then,  $i + \frac{x}{2}$  is the beginning position of an Abelian square of length  $2d$  iff  $w[\text{beg}(L_i)] = c_p = w[\text{end}(R_i) + 1]$ ,  $w[\text{beg}(R_i)] = c_q$ . Also, this is the only Abelian square of length  $2d$  beginning at positions between  $i$  and  $\text{bp}(i)$ .

*Proof.* ( $\Leftarrow$ ) Since  $w[\text{beg}(L_i)] = c_p = w[\text{end}(R_i) + 1]$  and  $w[\text{beg}(R_i)] = c_q$ , we have that  $\mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] - 2z = \mathcal{P}_{L_{i+z}}[p] - \mathcal{P}_{R_{i+z}}[p]$  and  $\mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] + 2z = \mathcal{P}_{R_{i+z}}[q] - \mathcal{P}_{L_{i+z}}[q]$  for any  $1 \leq z \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Since  $\frac{x}{2} \leq \min\{D_1^i, D_2^i, D_3^i\}$ , the Parikh vectors of the sliding windows become equal at position  $i + \frac{x}{2}$ .

( $\Rightarrow$ ) Since  $x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q] > 0$ ,  $\mathcal{P}_{L_{i+\frac{x}{2}}}[p] = \mathcal{P}_{L_{i+\frac{x}{2}}}[q]$ , and  $\mathcal{P}_{L_{i+\frac{x}{2}}}[q] = \mathcal{P}_{L_{i+\frac{x}{2}}}[p]$ , we have  $w[\text{beg}(L_i)] = c_p = w[\text{end}(R_i) + 1]$  and  $w[\text{beg}(R_i)] = c_q$ .

From the above arguments, it is clear that  $i + \frac{x}{2}$  is the only position between  $i$  and  $\text{bp}(i)$  where an Abelian square of length  $2d$  can start.  $\square$

**Lemma 4.** Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) = 3$ . Let  $c_p = w[\text{beg}(L_i)]$ ,  $c_{p'} = w[\text{end}(R_i) + 1]$ , and  $c_q = w[\text{beg}(R_i)]$ . Then,  $i + x$  with  $i < i + x \leq \text{bp}(i)$  is the beginning position of an Abelian square of length  $2d$  iff  $0 < x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{L_i}[p'] - \mathcal{P}_{R_i}[p'] = \frac{\mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q]}{2} \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Also, this is the only Abelian square of length  $2d$  beginning at positions between  $i$  and  $\text{bp}(i)$ .

*Proof.* ( $\Leftarrow$ ) Since  $w[\text{beg}(L_i)] = c_p$ ,  $w[\text{end}(R_i) + 1] = c_{p'}$  and  $w[\text{beg}(R_i)] = c_q$ , we have that  $\mathcal{P}_{L_i}[p] - z = \mathcal{P}_{L_{i+z}}[p]$ ,  $\mathcal{P}_{L_i}[q] + z = \mathcal{P}_{L_{i+z}}[q]$ ,  $\mathcal{P}_{R_i}[q] - z = \mathcal{P}_{R_{i+z}}[q]$ ,  $\mathcal{P}_{L_i}[q] + z = \mathcal{P}_{L_{i+z}}[q]$  and  $\mathcal{P}_{R_i}[p'] + z = \mathcal{P}_{R_{i+z}}[p']$  for any  $1 \leq z \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Since  $x \leq \min\{D_1^i, D_2^i, D_3^i\}$ , the Parikh vectors of the sliding windows become equal at position  $i + x$  and  $i < i + x \leq \text{bp}(i)$ .

( $\Rightarrow$ ) Since  $i < i + x \leq \text{bp}(i)$ , we have  $0 < x \leq \min\{D_1^i, D_2^i, D_3^i\}$ . Since  $w[\text{beg}(L_i)] = c_p$ ,  $w[\text{end}(R_i) + 1] = c_{p'}$ ,  $w[\text{beg}(R_i)] = c_q$ , and  $\mathcal{P}_{L_{i+x}} = \mathcal{P}_{R_{i+x}}$ , we have  $x = \mathcal{P}_{L_i}[p] - \mathcal{P}_{R_i}[p] = \mathcal{P}_{L_i}[p'] - \mathcal{P}_{R_i}[p'] = \frac{\mathcal{P}_{R_i}[q] - \mathcal{P}_{L_i}[q]}{2}$ .

From the above arguments, it is clear that  $i + x$  is the only position between  $i$  and  $\text{bp}(i)$  where an Abelian square of length  $2d$  can start.  $\square$

**Lemma 5.** Assume  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) \geq 4$ . Then, there exists no Abelian square of length  $2d$  beginning at any position  $j$  with  $i < j \leq \text{bp}(i)$ .

*Proof.* By the definition of  $\text{bp}(i)$ , we have that  $w[\text{beg}(L_i)] = w[\text{beg}(L_{\text{bp}(i)})]$ ,  $w[\text{beg}(R_i)] = w[\text{beg}(R_{\text{bp}(i)})]$ , and  $w[\text{end}(R_i)] = w[\text{end}(R_{\text{bp}(i)})]$ . Since the ending position of the left sliding window is adjacent to the beginning position of the right sliding window, we have  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) - \text{diff}(\mathcal{P}_{L_j}, \mathcal{P}_{R_j}) \leq 3$  for any  $i \leq j \leq \text{bp}(i)$ . Since we have assumed  $\text{diff}(\mathcal{P}_{L_i}, \mathcal{P}_{R_i}) \geq 4$ , we get  $\text{diff}(\mathcal{P}_{L_j}, \mathcal{P}_{R_j}) \geq 1$ . Thus there exist no Abelian squares starting at position  $j$ .  $\square$

## A.2 Proofs for lemmas in Section 4

**Lemma 7.** Assume  $c_{p_l} = c_{q_l} \neq c_{p_r} = c_{q_r}$ . There is a common Abelian common factor ( $w_1[i+x..i+x+d-1]$ ,  $w_2[k+y..k+y+d-1]$ ) of length  $d$  iff  $0 \leq x \leq \text{bp}_1(i) - i$ ,  $0 \leq y \leq \text{bp}_2(k) - k$  and  $x - y = \max(p_l) - \max(q_l) = \min(q_r) - \min(p_r)$ .

*Proof.* During the slide of the window on  $w_1$ , the number of occurrences of  $c_{p_l}$  decreases and that of  $c_{p_r}$  increases. That is,  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \max(p_l) - x$  and  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \min(p_r) + x$ . On the other hand, during the slide of the window on  $w_2$ , the number of occurrence of  $c_{q_l}$  decreases and that of  $c_{q_r}$  increases. That is,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y = \max(q_l) - y$  and  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[p_r] = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y = \min(q_r) + y$ .

Assume a pair  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  is a common Abelian factor of length  $d$ . Then,  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l]$  and  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r]$ , that is,  $\max(p_l) - x = \max(q_l) - y$  and  $\min(p_r) + x = \min(q_r) + y$ . Therefore  $x - y = \max(p_l) - \max(q_l) = \min(q_r) - \min(p_r)$ .

Assume that  $x - y = \max(p_l) - \max(q_l) = \min(q_r) - \min(p_r)$ . Then, we have that  $\max(p_l) - \max(q_l) = \mathcal{P}_{w_1[i..i+d-1]}[p_l] - \mathcal{P}_{w_2[k..k+d-1]}[q_l] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] + x - \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] - y = x - y$ , that is,  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l]$ . Also, we have that  $\min(q_r) - \min(p_r) = \mathcal{P}_{w_2[k..k+d-1]}[q_r] - \mathcal{P}_{w_1[i..i+d-1]}[p_r] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] - y - \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] + x = x - y$ , that is,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r]$ . Therefore, a pair  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  is a common Abelian factor of  $w_1$  and  $w_2$ .  $\square$

**Lemma 8.** Assume  $c_{p_r} \neq c_{p_l} = c_{q_l} \neq c_{q_r}$  and  $c_{p_r} \neq c_{q_r}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  iff  $\text{bp}_1(i) - i \geq x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \geq 0$ ,  $\text{bp}_2(k) - k \geq y = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \geq 0$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y$ .

*Proof.* During the slides of the windows on  $w_1$  and  $w_2$ , the numbers of occurrences of  $c_{q_r}$  in  $w_1$  and  $c_{p_r}$  in  $w_2$  do not change.

Assume there is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  of length  $d$ . Clearly  $0 \leq x \leq \text{bp}_1(i) - i$  and  $0 \leq y \leq \text{bp}_2(k) - k$ . Then, we have  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[p_r]$ ,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[q_r]$  and  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l]$ , that is,  $\min(p_r) + x = \mathcal{P}_{w_2[k..k+d-1]}[p_r]$ ,  $\min(q_r) + y = \mathcal{P}_{w_1[i..i+d-1]}[q_r]$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y$ . Consequently, we obtain  $x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r)$  and  $y = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r)$ .

Assume that  $\text{bp}_1(i) - i \geq x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \geq 0$ ,  $\text{bp}_2(k) - k \geq y = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \geq 0$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y$ . Then, we have that  $\min(p_r) + x = \mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[p_r]$ ,  $\min(q_r) + y = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[q_r]$  and  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l]$ . Therefore, a pair  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  is a common Abelian factor of  $w_1$  and  $w_2$ .  $\square$

**Lemma 9.** Assume  $c_{p_l} \neq c_{p_r} = c_{q_r} \neq c_{q_l}$  and  $c_{p_l} \neq c_{q_l}$ . There is a common Abelian factor  $(w_1[i+x..i+x+d-1], w_2[k+y..k+y+d-1])$  iff  $x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] \geq 0$ ,  $y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] \geq 0$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y$ .



Lemma 9 can be proved by a similar argument to the proof of Lemma 8.

**Lemma 10.** Assume  $c_{p_l} = c_{q_r} \neq c_{p_r} = c_{q_l}$ . There is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  iff  $x + y = \max(q_l) - \min(p_r) = \max(p_l) - \min(q_r)$ ,  $0 \leq x \leq \text{bp}_1(i) - i$  and  $0 \leq y \leq \text{bp}_2(k) - k$ .

*Proof.* When the window on  $w_1$  slides by  $x$  positions, the occurrence of  $c_{p_l}$  in the window decreases by  $x$  and the occurrence of  $c_{p_r}$  in the window increases by  $x$ . When the window on  $w_2$  slides by  $y$  positions, the occurrence of  $c_{q_l}$  in the window decreases by  $y$  and the occurrence of  $c_{q_r}$  in the window increases by  $y$ .

Assume there is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$ . Then  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \min(p_r) + x$ ,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y = \max(q_l) - y$ ,  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \max(p_l) - x$  and  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y = \min(q_r) + y$ . Therefore  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] \Leftrightarrow x + y = \max(q_l) - \min(p_r)$  and  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] \Leftrightarrow x + y = \max(p_l) - \min(q_r)$ .

Assume  $x + y = \max(q_l) - \min(p_r) = \max(p_l) - \min(q_r)$ . Clearly  $0 \leq x \leq \text{bp}_1(i) - i$  and  $0 \leq y \leq \text{bp}_2(k) - k$ . Then  $\max(q_l) - y = \min(p_r) + x$  and  $\max(p_l) - x = \min(q_r) + y$ , that is,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r]$  and  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r]$ . Therefore a pair  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  is a common Abelian factor of  $w_1$  and  $w_2$ .  $\square$

**Lemma 11.** Assume  $c_{p_l}$ ,  $c_{p_r}$ ,  $c_{q_l}$  and  $c_{q_r}$  are mutually distinct. There is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  iff  $0 \leq x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \leq \text{bp}_1(i) - i$  and  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$ .

*Proof.* During the slides, the numbers of occurrences of  $c_{q_l}$  and  $c_{q_r}$  in the window on  $w_1$  do not change, and those of  $c_{p_l}$  and  $c_{p_r}$  in the window on  $w_2$  do not change.

Assume there is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$ . Then,  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \min(p_r) + x = \mathcal{P}_{w_2[k..k+d-1]}[p_r]$ ,  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \max(p_l) - x = \mathcal{P}_{w_2[k..k+d-1]}[p_l]$ ,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \min(q_r) + y = \mathcal{P}_{w_1[i..i+d-1]}[q_r]$  and  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] = \max(q_l) - y = \mathcal{P}_{w_1[i..i+d-1]}[q_l] \Leftrightarrow 0 \leq x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \leq \text{bp}_1(i) - i$  and  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$ .

Assume  $0 \leq x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \leq \text{bp}_1(i) - i$  and  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$ . Then,  $x = \mathcal{P}_{w_1[i..i+d-1]}[p_l] - \mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \mathcal{P}_{w_2[k..k+d-1]}[q_r]$  and  $y = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \mathcal{P}_{w_2[k..k+d-1]}[q_r]$ . That is,  $\mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[p_l] = \mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l]$ ,  $\mathcal{P}_{w_2[k..k+d-1]}[p_r] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[p_r] = \mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r]$ ,  $\mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[q_l] = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l]$ , and  $\mathcal{P}_{w_1[i..i+d-1]}[q_r] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[q_r] = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r]$ . Therefore, a pair  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  is a common Abelian factor of  $w_1$  and  $w_2$ .  $\square$

**Lemma 12.** Assume  $c_{p_l} = c_{p_r}$  and  $c_{q_l} \neq c_{q_r}$ . There is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  of length  $d$  iff  $0 \leq x \leq \text{bp}_1(i) - i$ ,  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_l]$ .

*Proof.* During the slide, the number of occurrences of  $c_{p_l}$  ( $= c_{p_r}$ ) in the window on  $w_1$  does not change.

Assume that there is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$ . Clearly  $0 \leq x \leq \text{bp}_1(i) - i$  and  $0 \leq y \leq \text{bp}_2(k) - k$ . Then, it holds that  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] = \max(q_l) - y = \mathcal{P}_{w_1[i+x..i+x+d-1]}[q_l]$ ,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \min(q_r) + y = \mathcal{P}_{w_1[i+x..i+x+d-1]}[q_r]$  and  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[p_l] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l]$ , that is,  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r)$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_l]$ .

Assume  $y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r)$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_l]$ . Then,  $\mathcal{P}_{w_2[k..k+d-1]}[q_l] - y = \mathcal{P}_{w_1[i..i+d-1]}[q_l]$  and  $\mathcal{P}_{w_2[k..k+d-1]}[q_r] + y = \mathcal{P}_{w_1[i..i+d-1]}[q_r]$ , that is,  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] = \mathcal{P}_{w_1[i..i+d-1]}[q_l]$  and  $\mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \mathcal{P}_{w_1[i..i+d-1]}[q_r]$ . Therefore, a pair  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  is a common Abelian factor of length  $d$  of  $w_1$  and  $w_2$ .  $\square$

**Lemma 13.** Assume  $c_{q_l} = c_{q_r}$  and  $c_{p_l} \neq c_{p_r}$ . There is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  of length  $d$  iff  $0 \leq y \leq \text{bp}_2(k) - k$ ,  $x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \geq 0$  and  $\mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_2[k..k+d-1]}[q_l]$ .

Lemma 13 can be proved by a similar argument to the proof of Lemma 12.

**Lemma 14.** Assume  $c_{p_r} \neq c_{p_l} = c_{q_r} \neq c_{q_l}$  and  $c_{p_r} \neq c_{q_l}$ . There is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  of length  $d$  iff  $0 \leq x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r) \leq \text{bp}_1(i) - i$ ,  $0 \leq y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l] \leq \text{bp}_2(k) - k$  and  $x + y = \max(p_l) - \min(q_r)$ .

*Proof.* During the slides of the windows, the number of occurrences of  $c_{q_l}$  in the window on  $w_1$  and that of  $c_{p_r}$  in the window on  $w_2$  do not change.

Assume there is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$ . Then,  $\mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l] = \max(q_l) - y$ ,  $\mathcal{P}_{w_2[k..k+d-1]}[p_r] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r] = \min(p_r) + x$ ,  $\mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \min(p_l) + x = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r] = \max(q_r) - y$ , that is,  $y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l]$ ,  $x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r)$  and  $x + y = \max(p_l) - \min(q_r)$ .

Assume  $y = \max(q_l) - \mathcal{P}_{w_1[i..i+d-1]}[q_l]$ ,  $x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \min(p_r)$  and  $x + y = \max(p_l) - \min(q_r)$ . Then,  $y = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - \mathcal{P}_{w_1[i..i+d-1]}[q_l]$ ,  $x = \mathcal{P}_{w_2[k..k+d-1]}[p_r] - \mathcal{P}_{w_1[i..i+d-1]}[p_r]$  and  $x + y = \mathcal{P}_{w_1[i..i+d-1]}[p_l] - \mathcal{P}_{w_2[k..k+d-1]}[q_r]$ , that is,  $\mathcal{P}_{w_1[i..i+d-1]}[q_l] = \mathcal{P}_{w_1[i+x..i+x+d-1]}[q_l] = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_l]$ ,  $\mathcal{P}_{w_2[k..k+d-1]}[p_r] = \mathcal{P}_{w_2[k+y..k+y+d-1]}[p_r] = \mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_r]$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_l] - x = \mathcal{P}_{w_1[i+x..i+x+d-1]}[p_l] = \mathcal{P}_{w_2[k..k+d-1]}[q_r] + y = \mathcal{P}_{w_2[k+y..k+y+d-1]}[q_r]$ . Therefore, a pair  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  is a common Abelian factor of length  $d$  of  $w_1$  and  $w_2$ .  $\square$

**Lemma 15.** *Assume  $c_{p_l} \neq c_{q_l} = c_{p_r} \neq c_{q_r}$  and  $c_{p_l} \neq c_{q_r}$ . There is a common Abelian factor  $(w_1[i + x..i + x + d - 1], w_2[k + y..k + y + d - 1])$  of length  $d$  iff  $0 \leq x = \max(p_l) - \mathcal{P}_{w_2[k..k+d-1]}[p_l] \leq \text{bp}_1(i) - i$ ,  $0 \leq y = \mathcal{P}_{w_1[i..i+d-1]}[q_r] - \min(q_r) \leq \text{bp}_2(k) - k$  and  $\mathcal{P}_{w_1[i..i+d-1]}[p_r] + x = \mathcal{P}_{w_2[k..k+d-1]}[q_l] - y$ .*

Lemma 15 can be proved by a similar argument to the proof of Lemma 14.

## B Appendix: Examples

### B.1 Example for Computing Abelian squares using RLEs

Here we show some examples on how our algorithm of Section 4 computes all Abelian squares of a given string based on its RLE.

Consider string  $w = a^{12}b^4a^3c^2d^2c^2a^2$  over alphabet  $\Sigma = \{a, b, c, d\}$  of size 4. Let  $d = 4$ .

$\begin{array}{cccccccccccccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 \\ a & a & a & a & a & a & a & a & a & a & a & b & b & b & b & a & a & a & c & c & d & d & c & a & c & c & \end{array}$

**Fig. 7.**  $beg(L_1) = 1, beg(R_1) = 5, end(R_1) + 1 = 9, w[beg(L_1)] = w[beg(R_1)] = w[end(R_1) + 1] = a$ .

See Fig. 7 for the initial step of our algorithm, where  $i = 1$ . As  $diff(\mathcal{P}_{L_1}, \mathcal{P}_{R_1}) = 0$ ,  $w[1..8] = aaaaaaaaa$  is an Abelian square. Since  $\min\{D_1^1, D_2^1, D_3^1\} = \min\{12, 8, 4\} = 4$ , the next break point is  $bp(1) = 1 + 4 = 5$ . Since  $w[beg(L_1)] = w[beg(R_1)] = w[end(R_1) + 1] = a$  and it follows from Lemma 1 that the substrings of length  $2d = 8$  between 1 and the break point are all equal, i.e.,  $w[1..8] = w[2..9] = w[3..10] = w[4..11] = w[5..12]$ , and all of them are Abelian squares. Hence we output a triple  $\langle 1, 5, 4 \rangle$  representing all these Abelian squares. We update  $i \leftarrow bp(1) = 5$ , and proceed to the next step.

$\begin{array}{cccccccccccccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 \\ a & a & a & a & a & a & a & a & a & b & b & b & b & a & a & a & c & c & d & d & c & a & c & c & \end{array}$

**Fig. 8.**  $beg(L_5) = 5, beg(R_5) = 9, end(R_5) + 1 = 13, w[beg(L_5)] = w[beg(R_5)] = a, w[end(R_5) + 1] = b$ .

Next, see Fig. 8 where the left window has been shifted to  $L_5 = w[5..6] = aaaa$  and the right window has been shifted to  $R_5 = w[8..12] = aaaaa$ . Since  $\min\{D_1^5, D_2^5, D_3^5\} = \min\{8, 4, 4\} = 4$ , the next break point is  $bp(5) = 5 + 4 = 9$ . Since  $\mathcal{P}_{L_5} = \mathcal{P}_{R_5}$  and  $w[beg(L_5)] = w[beg(R_5)] = a \neq w[end(R_5) + 1] = b$ , it follows from Lemma 1 that there are no Abelian squares between 5 and the break point 9. We update  $i \leftarrow bp(5) = 9$ , and proceed to the next step.

Next, see Fig. 9 where the left window has been shifted to  $L_9 = w[9..12] = aaaa$  and the right window has been shifted to  $R_9 = w[13..16] = bbbb$ . Since  $\min\{D_1^9, D_2^9, D_3^9\} = \min\{4, 4, 3\} = 3$ , the next break point is  $bp(9) = 9 + 3 = 12$ . Since  $diff(\mathcal{P}_{L_9}, \mathcal{P}_{R_9}) = 2$ ,  $w[beg(L_9)] = w[end(R_9) + 1] = a \neq w[beg(R_9)] = b$ , and  $\frac{\mathcal{P}_{L_9}[a] - \mathcal{P}_{R_9}[a] = \mathcal{P}_{R_9}[b] - \mathcal{P}_{L_9}[b]}{2} = 2 \leq \min\{D_1^9, D_2^9, D_3^9\} = 3$ , it follows from

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 9.**  $beg(L_9) = 9, beg(R_9) = 13, end(R_9) + 1 = 17, w[beg(L_9)] = a, w[beg(R_9)] = b, w[end(R_9) + 1] = a$ .

Lemma 3 that  $w[11..18]$  is the only Abelian square of length  $2d = 8$  starting at positions between 9 and 12. We hence output  $\langle 11, 11, 4 \rangle$ . We update  $i \leftarrow bp(9) = 12$ , and proceed to the next step.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 10.**  $beg(L_{12}) = 12, beg(R_{12}) = 16, end(R_{12}) + 1 = 20, w[beg(L_{12})] = a, w[beg(R_{12})] = b, w[end(R_{12}) + 1] = c$ .

Next, see Fig. 10 where the left window has been shifted to  $L_{12} = w[12..15] = abbb$  and the right window has been shifted to  $R_{12} = w[16..19] = baaa$ . Since  $\min\{D_1^{12}, D_2^{12}, D_3^{12}\} = \min\{1, 1, 1\} = 1$ , the next break point is  $bp(12) = 12 + 1 = 13$ . Since  $diff(\mathcal{P}_{L_{12}}, \mathcal{P}_{R_{12}}) = 3$  and  $w[beg(L_{12})] = a \neq w[beg(R_{12})] = b \neq w[end(R_{12}) + 1] = c$ , it follows from Lemma 2 and Lemma 3 that there are no Abelian squares starting at positions between 12 and 13. We update  $i \leftarrow bp(12) = 13$ , and proceed to the next step.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 11.**  $beg(L_{13}) = 13, beg(R_{13}) = 17, end(R_{13}) + 1 = 21, w[beg(L_{13})] = b, w[beg(R_{13})] = a, w[end(R_{13}) + 1] = c$ .

Next, see Fig. 11 where the left window has been shifted to  $L_{13} = w[13..16] = bbbb$  and the right window has been shifted to  $R_{13} = w[17..20] = aaac$ . Since  $\min\{D_1^{13}, D_2^{13}, D_3^{13}\} = \min\{4, 3, 1\} = 1$ , the next break point is  $bp(13) = 13 + 1 = 14$ . Since  $diff(\mathcal{P}_{L_{13}}, \mathcal{P}_{R_{13}}) = 3$  and  $\mathcal{P}_{L_{13}}[b] - \mathcal{P}_{R_{13}}[b] = 4 \neq -1 = \mathcal{P}_{L_{13}}[c] - \mathcal{P}_{R_{13}}[c]$ , it follows from Lemma 4 that 14 is not the beginning position of an Abelian square of length  $2d = 8$ . We update  $i \leftarrow bp(13) = 14$ , and proceed to the next step.

Next, see Fig. 12 where the left window has been shifted to  $L_{14} = w[14..17] = bbba$  and the right window has been shifted to  $R_{14} = w[18..21] = aacc$ . Since  $\min\{D_1^{14}, D_2^{14}, D_3^{14}\} = \min\{3, 2, 2\} = 2$ , the next break point is  $bp(14) = 14 +$

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 12.**  $beg(L_{14}) = 14, beg(R_{14}) = 18, end(R_{14}) + 1 = 122, w[beg(L_{14})] = b, w[beg(R_{14})] = a, w[end(R_{14}) + 1] = d.$

2 = 16. Since  $diff(\mathcal{P}_{L_{14}}, \mathcal{P}_{R_{14}}) = 3$  and  $\mathcal{P}_{L_{14}}[b] - \mathcal{P}_{R_{14}}[b] = 3 \neq -1 = \mathcal{P}_{L_{14}}[c] - \mathcal{P}_{R_{14}}[c]$ , it follows from Lemma 4 that there are no Abelian squares starting at positions between 14 and 16. We update  $i \leftarrow bp(14) = 16$ , and proceed to the next step.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 13.**  $beg(L_{16}) = 16, beg(R_{16}) = 20, end(R_{16}) + 1 = 24, w[beg(L_{16})] = b, w[beg(R_{16})] = w[end(R_{16}) + 1] = c$

Next, see Fig. 13 where the left window has been shifted to  $L_{16} = w[16..19] = baaa$  and the right window has been shifted to  $R_{16} = w[20..23] = ccdd$ . Since  $\min\{D_1^{16}, D_2^{16}, D_3^{16}\} = \min\{1, 2, 2\} = 1$ , the next break point is  $bp(16) = 16 + 1 = 17$ . Since  $diff(\mathcal{P}_{L_{16}}, \mathcal{P}_{R_{16}}) = 3$  and  $\mathcal{P}_{L_{16}}[b] - \mathcal{P}_{R_{16}}[b] = 1 \neq -2 = \mathcal{P}_{L_{16}}[c] - \mathcal{P}_{R_{16}}[c]$ , it follows from Lemma 4 that 16 is not the beginning position of an Abelian square of length  $2d = 8$ . We update  $i \leftarrow bp(16) = 17$ , and proceed to the next step.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 14.**  $beg(L_{17}) = 17, beg(R_{17}) = 21, end(R_{17}) + 1 = 25, w[beg(L_{17})] = a, w[beg(R_{17})] = c, w[end(R_{17}) + 1] = a$

Next, see Fig. 14 where the left window has been shifted to  $L_{17} = w[17..20] = aaac$  and the right window has been shifted to  $R_{17} = w[21..24] = cddc$ . Since  $\min\{D_1^{17}, D_2^{17}, D_3^{17}\} = \min\{3, 1, 1\} = 1$ , the next break point is  $bp(17) = 17 + 1 = 18$ . Since  $diff(\mathcal{P}_{L_{17}}, \mathcal{P}_{R_{17}}) = 3$  and  $\mathcal{P}_{L_{17}}[a] - \mathcal{P}_{R_{17}}[a] = 3 \neq -2 = \mathcal{P}_{L_{17}}[c] - \mathcal{P}_{R_{17}}[c]$ , it follows from Lemma 4 that 17 is not the beginning position of an Abelian square of length  $2d = 8$ . We update  $i \leftarrow bp(17) = 18$ , and proceed to the next step.

Next, see Fig. 15 where the left window has been shifted to  $L_{18} = w[18..21] = aacc$  and the right window has been shifted to  $R_{18} = w[20..25] = ddcc$ . Since

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 15.**  $beg(L_{18}) = 18, beg(R_{18}) = 22, end(R_{18}) + 1 = 26, w[beg(L_{18})] = a, w[beg(R_{18})] = d, w[end(R_{18}) + 1] = c$

$\min\{D_1^{18}, D_2^{18}, D_3^{18}\} = \min\{2, 2, 2\} = 2$ , the next break point is  $bp(18) = 18 + 2 = 20$ . Since  $diff(\mathcal{P}_{L_{18}}, \mathcal{P}_{R_{18}}) = 3$ , we use Lemma 4. Since  $\mathcal{P}_{L_{18}}[a] - \mathcal{P}_{R_{18}}[a] = \mathcal{P}_{L_{18}}[c] - \mathcal{P}_{R_{18}}[c] = \frac{\mathcal{P}_{R_{18}}[d] - \mathcal{P}_{L_{18}}[d]}{2} = 1 \leq \min\{D_1^{18}, D_2^{18}, D_3^{18}\} = 2$ , it follows from Lemma 4 that  $w[19..26]$  is an Abelian square of length  $2d = 8$ . We hence output  $\langle 19, 19, 4 \rangle$ . We update  $i \leftarrow bp(19) = 20$ , and proceed to the next step.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
a a a a a a a a a a a b b b b a a a c c d d c a c c

**Fig. 16.**  $beg(L_{20}) = 20, beg(R_{20}) = 24, w[beg(L_{20})] = c, w[beg(R_{20})] = c$

Next, see Fig. 16 where the left window has been shifted to  $L_{20} = w[20..23] = ccdd$  and the right window has been shifted to  $R_{20} = w[24..27] = cacc$ . Since  $diff(\mathcal{P}_{L_{20}}, \mathcal{P}_{R_{20}}) = 3$  the right end of the right window has reached the last positions of the input string, the algorithm terminates here. Recall that this algorithm computed all the Abelian squares of length  $2d = 8$  in this string.